

# Máster Avanzado de Programación en Python para Hacking, BigData y Machine Learning

FUNDAMENTOS DE PYTHON

# LECCIÓN 04

## DISTRIBUCIÓN DE PAQUETES. INTEGRACIÓN CON OTROS LENGUAJES.

# ÍNDICE

Introducción

Objetivos

Distribución de paquetes

Caso práctico para crear un paquete

Integración de Python con otros lenguajes de Programación

Conclusiones

# INTRODUCCIÓN

En esta lección aprenderemos a empaquetar nuestros programas implementados en el lenguaje de programación Python. La creación de estos paquetes es de utilidad para importarlos de una forma cómoda en futuras implementaciones que realicemos, así como distribuir nuestro código a la comunidad abierta de desarrolladores.

Por otro lado, estudiaremos como utilizar código procedente de otros lenguajes de programación como R en nuestras implementaciones de Python.

## OBJETIVOS

Al finalizar esta lección serás capaz de:

- 1 Conocer las herramientas de distribución de código Python.
- 2 Saber distribuir de forma pública y privada un proyecto Python.
- 3 Conocer las ventajas de integrar diferentes lenguajes de programación.
- 4 Saber integrar código R en Python.

## Distribución de paquetes

Decidir distribuir nuestro código de forma pública puede ser una opción muy interesante. De este modo podremos facilitar las tareas de otros desarrolladores, colaborando en la creación y mejora de los repositorios de Python.

Por ejemplo, imaginen el caso de la librería matplotlib, para realizar gráficos. Esta librería podemos instalarla por medio del gestor de paquetes pip, posteriormente la podemos importar y utilizar en nuestro código. Esta misma tarea es la que deseamos mostrar en esta lección

## Distribución de paquetes

Cada software que desarrollemos puede ser empaquetado para utilizarlo en otros proyectos propios o publicarlo en el Python Package Index (PyPI), de modo que nuestro software estará disponible para una gran comunidad de desarrolladores, ejecutando el comando:

```
pip install <nombre_paquete>
```

*Para poder empaquetar nuestro proyecto, éste debe seguir una estructura determinada.*

## Distribución de paquetes

### DIRECTORIO\_DEL\_PROYECTO

- LICENSE
- MANIFEST.in
- README.txt
- Setup.py

### NOMBRE\_DEL\_PAQUETE

- \_\_init\_\_.py
- ARCHIVO1.py
- ARCHIVO2.py
- ARCHIVO\_N.py



## Distribución de paquetes

Para comenzar a empaquetar nuestro código, necesitamos tener instaladas dos librerías:

**Wheel.** Herramienta utilizada para empaquetar el código. Para instalarla debemos ejecutar lo siguiente:

- `pip install wheel`

**Twine.** Herramienta para subir los proyectos a PyPI. Para instalarla debemos ejecutar lo siguiente:

- `pip install twine`

## Distribución de paquetes

PyPI es el repositorio de software oficial para aplicaciones de terceros en el lenguaje de programación Python.

Los desarrolladores de Python pretenden que sea un catálogo exhaustivo de todos los paquetes de Python escritos en código abierto.  
(<https://pypi.org/>)

## Caso práctico para crear un paquete

Una vez mostrada la base teórica sobre cómo empaquetar nuestro código, procedemos a mostrar un ejemplo práctico, para mostrar todo el proceso paso a paso. En este caso práctico, crearemos un módulo llamado Pyoperaciones.

El código se encuentra en un único fichero .py y contiene un conjunto de funciones para realizar operaciones básicas con números enteros, y que podremos utilizar en cualquier programa que importemos dicho módulo

## Caso práctico para crear un paquete

Existen diversas formas para crear un paquete. En nuestro caso particular utilizaremos la herramienta **wheel**, debido a su sencillez y rapidez. Para construir un paquete con dicha herramienta, debemos ejecutar el siguiente comando en una terminal:

- *python ./setup.py bdist\_wheel*

Una vez finalice el proceso, se habrá creado un nuevo directorio llamado “dist” que incluirá el paquete creado por **wheel** “nombre\_paquete-version-py3-none-any.whl”. Además, se creará un directorio adicional “build” con los archivos generados durante el proceso de generación.

## Caso práctico para crear un paquete

```
ramon@ramon-rd: ~/Escritorio/programaPython
ramon@ramon-rd:~/Escritorio/programaPython$ python setup.py bdist_wheel
running bdist_wheel
running build
running build_py
creating build
creating build/lib
creating build/lib/pyoperaciones
copying pyoperaciones/__init__.py -> build/lib/pyoperaciones
installing to build/bdist.linux-x86_64/wheel
running install
running install_lib
creating build/bdist.linux-x86_64
creating build/bdist.linux-x86_64/wheel
creating build/bdist.linux-x86_64/wheel/pyoperaciones
copying build/lib/pyoperaciones/__init__.py -> build/bdist.linux-x86_64/wheel/pyoperaciones
running install_egg_info
running egg_info
writing pyoperaciones.egg-info/PKG-INFO
writing dependency_links to pyoperaciones.egg-info/dependency_links.txt
writing top-level names to pyoperaciones.egg-info/top_level.txt
reading manifest file 'pyoperaciones.egg-info/SOURCES.txt'
reading manifest template 'MANIFEST.in'
warning: no files found matching 'LICENSE.txt'
warning: no files found matching 'examples/*'
warning: no files found matching 'examples/txt/*'
writing manifest file 'pyoperaciones.egg-info/SOURCES.txt'
copying pyoperaciones.egg-info to build/bdist.linux-x86_64/wheel/pyoperaciones-0.1.egg-info
running install_scripts
adding license file "LICENSE" (matched pattern "LICEN[CS]E*")
creating build/bdist.linux-x86_64/wheel/pyoperaciones-0.1.dist-info/WHEEL
creating 'dist/pyoperaciones-0.1-py3-none-any.whl' and adding 'build/bdist.linux-x86_64/wheel' to it
adding 'pyoperaciones/__init__.py'
adding 'pyoperaciones-0.1.dist-info/LICENSE'
adding 'pyoperaciones-0.1.dist-info/METADATA'
adding 'pyoperaciones-0.1.dist-info/WHEEL'
```

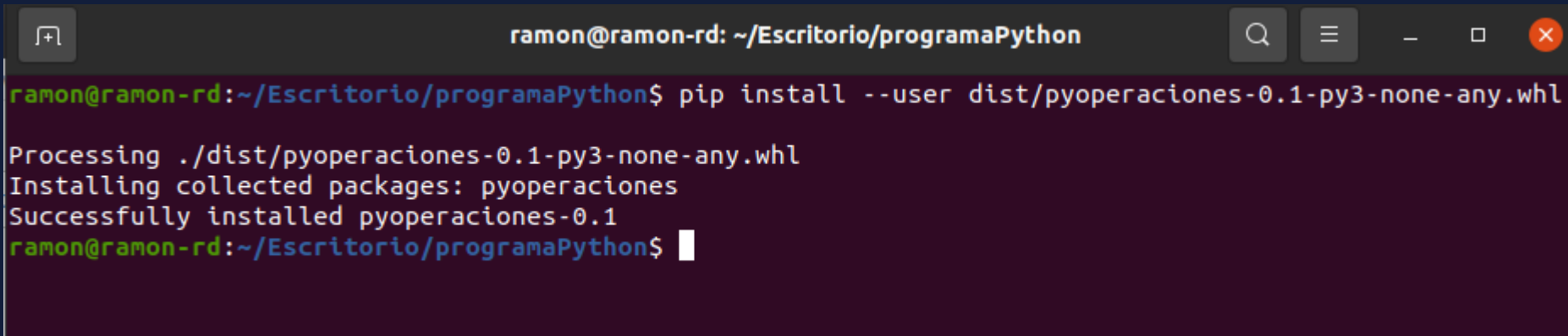
## Caso práctico para crear un paquete

Una vez que hemos creado nuestro paquete, tenemos dos opciones:

- Instalarlo en nuestra máquina para comprobar su correcto funcionamiento.
- Subir el paquete del proyecto a Pypi.

## Caso práctico para crear un paquete

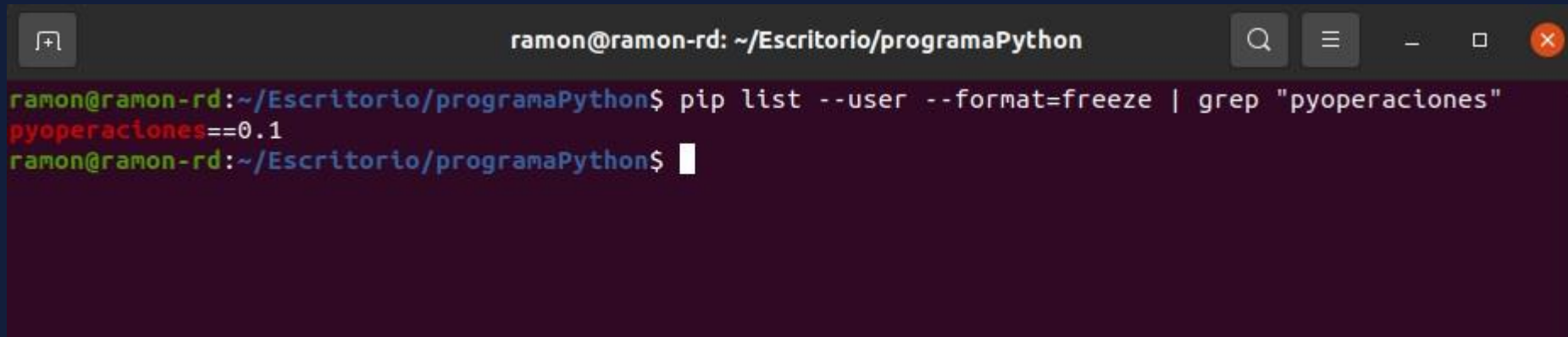
Caso 1: Instalación del paquete en nuestra máquina:

A terminal window with a dark background. The title bar shows 'ramon@ramon-rd: ~/Escritorio/programaPython'. The terminal text shows a command to install a wheel file, followed by confirmation messages from pip.

```
ramon@ramon-rd: ~/Escritorio/programaPython$ pip install --user dist/pyoperaciones-0.1-py3-none-any.whl
Processing ./dist/pyoperaciones-0.1-py3-none-any.whl
Installing collected packages: pyoperaciones
Successfully installed pyoperaciones-0.1
ramon@ramon-rd:~/Escritorio/programaPython$
```

## Caso práctico para crear un paquete

Para comprobar que el paquete ha sido instalado correctamente:

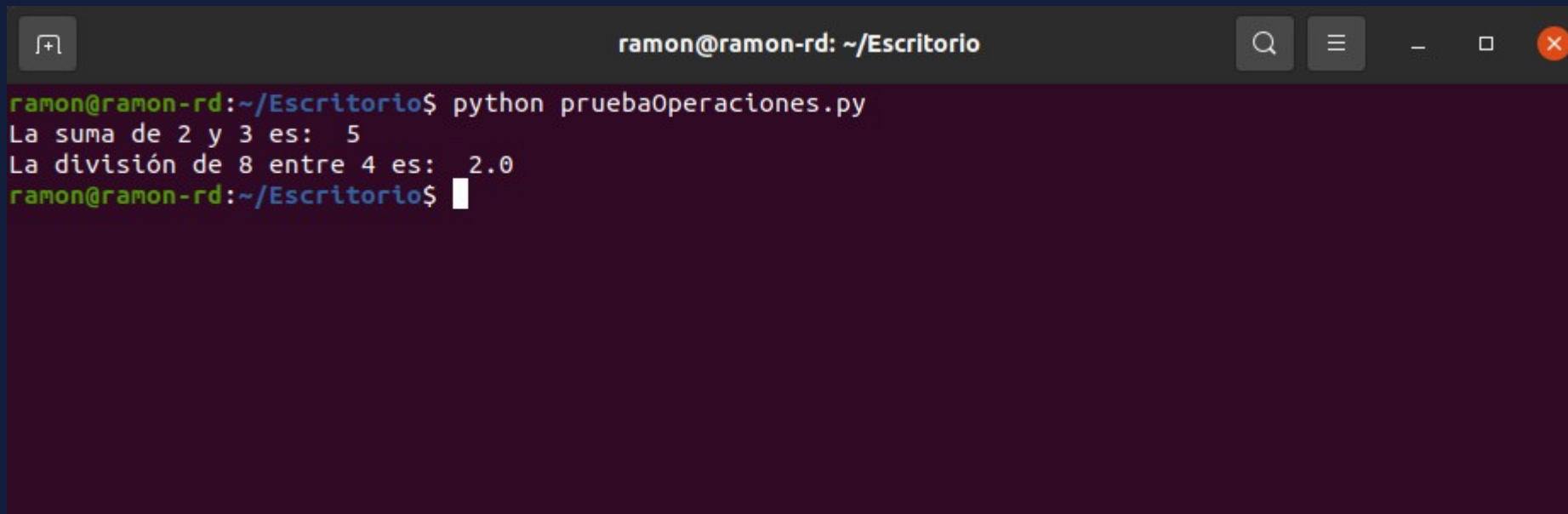


```
ramon@ramon-rd: ~/Escritorio/programaPython
ramon@ramon-rd:~/Escritorio/programaPython$ pip list --user --format=freeze | grep "pyoperaciones"
pyoperaciones==0.1
ramon@ramon-rd:~/Escritorio/programaPython$
```



## Caso práctico para crear un paquete

```
1 import pyoperaciones as po
2 |
3 print("La suma de 2 y 3 es: ", po.sumaEnteros(2,3))
4
5 print("La división de 8 entre 4 es: ", po.divideEnteros(8, 4))
```



A terminal window titled "ramon@ramon-rd: ~/Escritorio" with standard window controls. The prompt is "ramon@ramon-rd:~/Escritorio\$". The command "python pruebaOperaciones.py" has been executed, resulting in two lines of output: "La suma de 2 y 3 es: 5" and "La división de 8 entre 4 es: 2.0". The prompt is now "ramon@ramon-rd:~/Escritorio\$" with a cursor.

```
ramon@ramon-rd:~/Escritorio$ python pruebaOperaciones.py
La suma de 2 y 3 es: 5
La división de 8 entre 4 es: 2.0
ramon@ramon-rd:~/Escritorio$
```

## Caso práctico para crear un paquete

Para desinstalar el paquete:

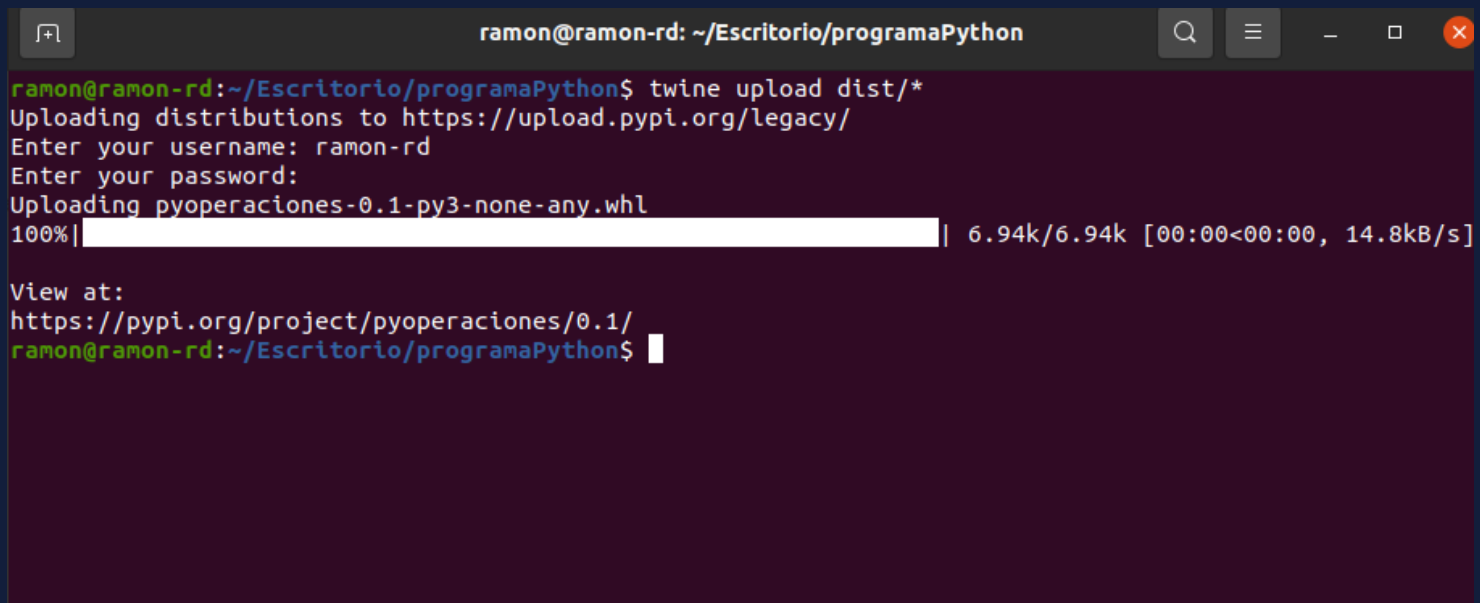


```
ramon@ramon-rd: ~/Escritorio
ramon@ramon-rd:~/Escritorio$ pip uninstall pyoperaciones
Found existing installation: pyoperaciones 0.1
Uninstalling pyoperaciones-0.1:
  Would remove:
    /home/ramon/.local/lib/python3.8/site-packages/pyoperaciones-0.1.dist-info/*
    /home/ramon/.local/lib/python3.8/site-packages/pyoperaciones/*
Proceed (y/n)? y
  Successfully uninstalled pyoperaciones-0.1
ramon@ramon-rd:~/Escritorio$
```

## Caso práctico para crear un paquete

Caso 2: subir el paquete del proyecto a PyPI:

- En primer lugar, debemos completar el formulario de registro de Pypi (<https://pypi.org/account/register/>).
- Subir el paquete a PyPI. Para ello, ejecutaremos en una terminal lo siguiente:



```
ramon@ramon-rd: ~/Escritorio/programaPython
ramon@ramon-rd:~/Escritorio/programaPython$ twine upload dist/*
Uploading distributions to https://upload.pypi.org/legacy/
Enter your username: ramon-rd
Enter your password:
Uploading pyoperaciones-0.1-py3-none-any.whl
100%|████████████████████████████████████████████████████████████████████████████████| 6.94k/6.94k [00:00<00:00, 14.8kB/s]

View at:
https://pypi.org/project/pyoperaciones/0.1/
ramon@ramon-rd:~/Escritorio/programaPython$
```

## Caso práctico para crear un paquete

Ahora podemos instalar nuestro paquete desde cualquier máquina por medio del gestor de paquetes pip.

```
ramon@ramon-rd:~/Escritorio/programaPython$ pip install pyoperaciones
Collecting pyoperaciones
  Downloading pyoperaciones-0.1-py3-none-any.whl (2.7 kB)
Installing collected packages: pyoperaciones
Successfully installed pyoperaciones-0.1
ramon@ramon-rd:~/Escritorio/programaPython$
```

## Integración de Python con otros lenguajes de Programación

Python ofrece la posibilidad de utilizar códigos implementados en otros lenguajes de programación dentro de un programa implementado en Python.

Podemos encontrar, por ejemplo, la librería rpy2 que nos permite integrar código realizado en R en nuestras implementaciones de Python.

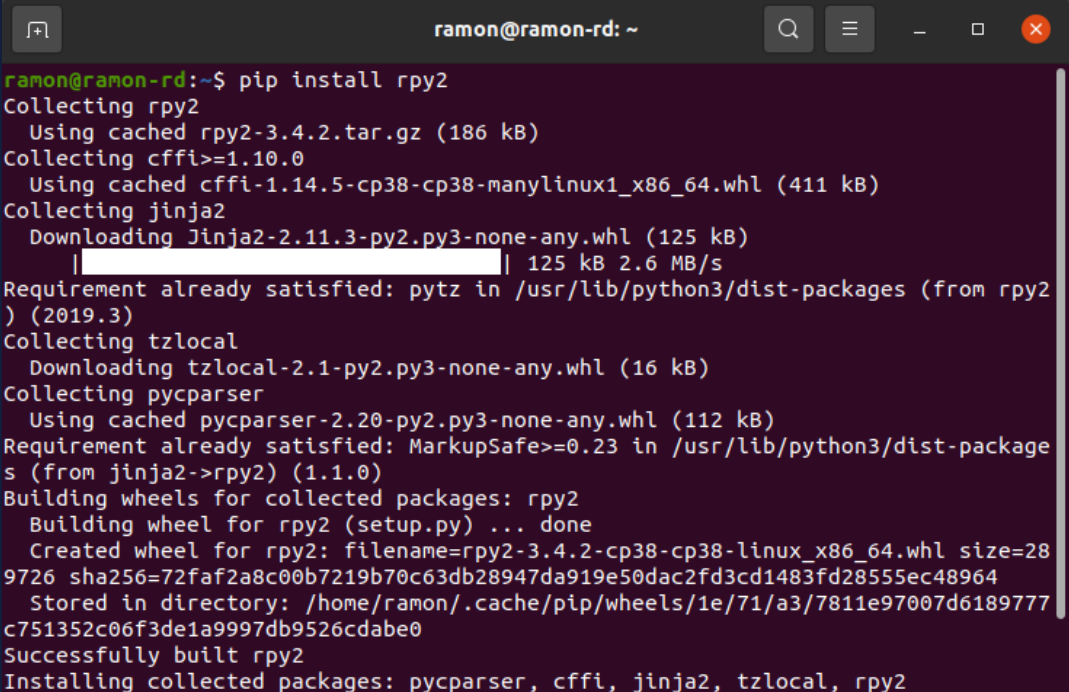
La integración de código nos permite explotar al máximo las ventajas de varios lenguajes de programación y unificarlos en un único programa.

Esta integración nos permitirá ahorrar tiempo de desarrollo, ya que podremos hacer uso de paquetes que hayan sido implementados en otro lenguaje de programación.

## Integración de Python con otros lenguajes de Programación

Para utilizar rpy2 necesitarás tener instalado tanto Python como R, además de las librerías R que quieras utilizar.

Para poder trabajar con R y Python de forma simultánea, debemos instalar el paquete rpy2 como sigue:



```
ramon@ramon-rd: ~  
ramon@ramon-rd:~$ pip install rpy2  
Collecting rpy2  
  Using cached rpy2-3.4.2.tar.gz (186 kB)  
Collecting cffi>=1.10.0  
  Using cached cffi-1.14.5-cp38-cp38-manylinux1_x86_64.whl (411 kB)  
Collecting jinja2  
  Downloading Jinja2-2.11.3-py2.py3-none-any.whl (125 kB)  
    |████████████████████| 125 kB 2.6 MB/s  
Requirement already satisfied: pytz in /usr/lib/python3/dist-packages (from rpy2) (2019.3)  
Collecting tzlocal  
  Downloading tzlocal-2.1-py2.py3-none-any.whl (16 kB)  
Collecting pycparser  
  Using cached pycparser-2.20-py2.py3-none-any.whl (112 kB)  
Requirement already satisfied: MarkupSafe>=0.23 in /usr/lib/python3/dist-packages (from jinja2->rpy2) (1.1.0)  
Building wheels for collected packages: rpy2  
  Building wheel for rpy2 (setup.py) ... done  
  Created wheel for rpy2: filename=rpy2-3.4.2-cp38-cp38-linux_x86_64.whl size=289726 sha256=72faf2a8c00b7219b70c63db28947da919e50dac2fd3cd1483fd28555ec48964  
  Stored in directory: /home/ramon/.cache/pip/wheels/1e/71/a3/7811e97007d6189777c751352c06f3de1a9997db9526cdabe0  
Successfully built rpy2  
Installing collected packages: pycparser, cffi, jinja2, tzlocal, rpy2
```

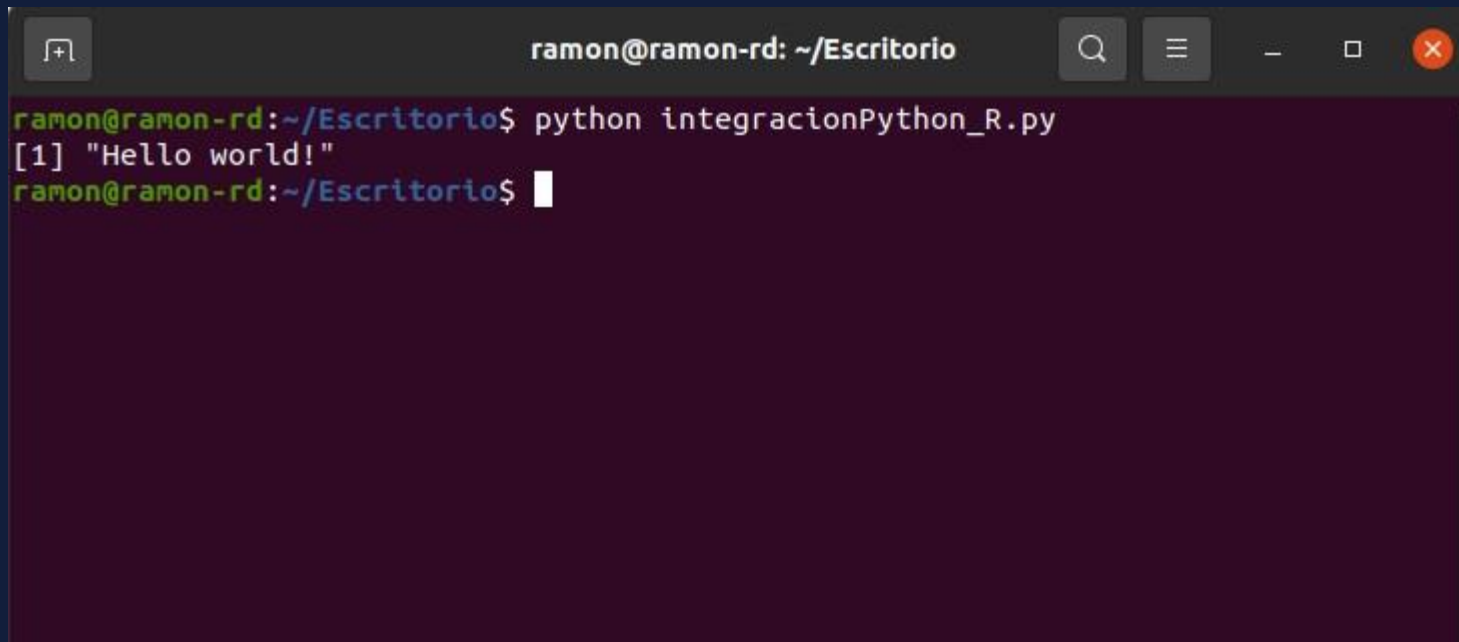
## Integración de Python con otros lenguajes de Programación

Una vez instalado, podemos crear un programa e importar la librería rpy2, que nos permitirá emplear utilizar código R dentro del código Python.

Pero antes de nada, ¿qué es rpy2? Rpy2 es una interfaz que permite que podamos comunicar información entre R y Python y que podamos acceder a las funcionalidades de R desde Python. Por lo tanto, podemos estar empleando Python para el desarrollo de todo nuestro programa, y en caso de que necesitemos emplear cualquier librería estadística de R, podremos acceder a misma usando rpy2.

## Integración de Python con otros lenguajes de Programación

```
from rpy2.robjects import r  
  
r('print("Hello world!")')
```



```
ramon@ramon-rd: ~/Escritorio  
ramon@ramon-rd:~/Escritorio$ python integracionPython_R.py  
[1] "Hello world!"  
ramon@ramon-rd:~/Escritorio$
```

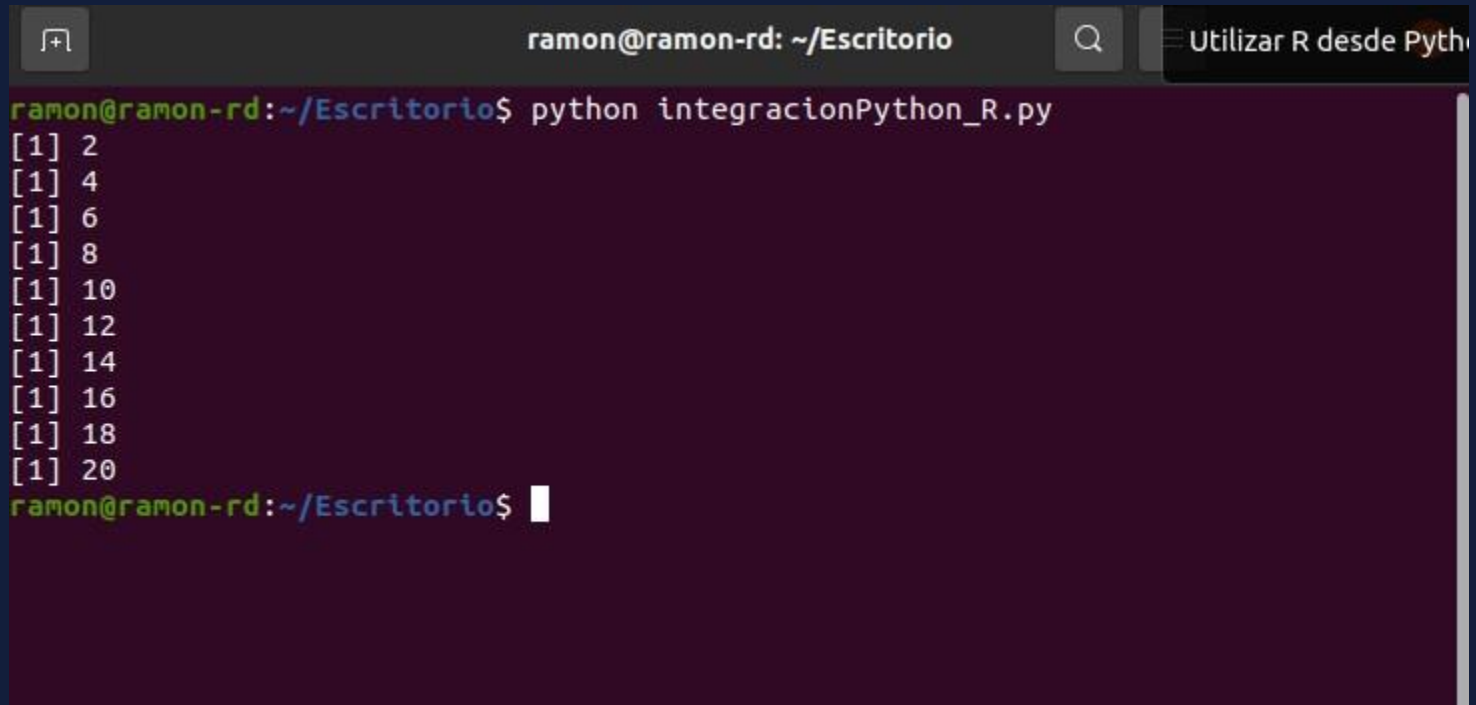


## Integración de Python con otros lenguajes de Programación

```
from rpy2.robjects import r

r('''
n <- 20
for (i in 1:n){
  if(i%%2 == 0){
    print(i)
  }
}
''')
```

## Integración de Python con otros lenguajes de Programación



```
ramon@ramon-rd: ~/Escritorio
ramon@ramon-rd:~/Escritorio$ python integracionPython_R.py
[1] 2
[1] 4
[1] 6
[1] 8
[1] 10
[1] 12
[1] 14
[1] 16
[1] 18
[1] 20
ramon@ramon-rd:~/Escritorio$
```

## Integración de Python con otros lenguajes de Programación

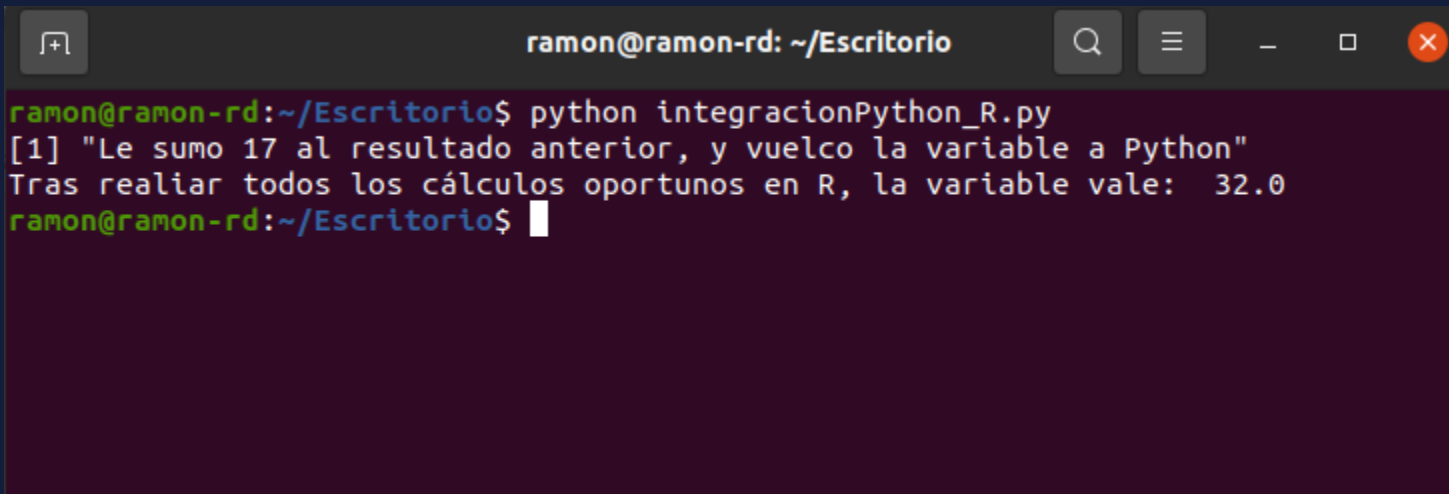
```
from rpy2.robjects import r

a = 5

r.assign('v', a)
r('z <- v*3')
r('sprintf("La variable vale ahora %i", | z)')
r('z <- z+17')
r('print("Le sumo 17 al resultado anterior, y vuelco la variable a Python")')

b = r('z')
print("Tras realizar todos los cálculos oportunos en R, la variable vale: " , b[0])
```

## Integración de Python con otros lenguajes de Programación



```
ramon@ramon-rd: ~/Escritorio
ramon@ramon-rd:~/Escritorio$ python integracionPython_R.py
[1] "Le sumo 17 al resultado anterior, y vuelco la variable a Python"
Tras realizar todos los cálculos oportunos en R, la variable vale: 32.0
ramon@ramon-rd:~/Escritorio$
```

## Integración de Python con otros lenguajes de Programación

```
import rpy2.robjects as ro

codigo_r = """
pares <- function(n){
  for (i in 1:n){
    if(i%%2 == 0){
      print(i)
    }
  }
}
"""
ro.r(codigo_r)

pares_py = ro.globalenv['pares']

pares_py(50)
```

## CONCLUSIONES

1

Empaquetar nuestro código nos permitirá portar de una forma más cómoda nuestras propias implementaciones y utilizarla en futuros desarrollos.

2

Distribuir de forma pública nuestros desarrollos mejora los repositorios de Python y nos permite colaborar con una gran comunidad de desarrolladores.

3

La integración de código de otros lenguajes de programación con Python permite explotar las ventajas de dos lenguajes de programación.



MUCHAS GRACIAS POR SU ATENCIÓN



[rrueda@grupomainjobs.com](mailto:rrueda@grupomainjobs.com)



Ramón Rueda Delgado

<https://www.linkedin.com/in/ramon-rueda/>



[twitter.com/eiposgrados](https://twitter.com/eiposgrados)



[facebook.com/eiposgrados](https://facebook.com/eiposgrados)



[instagram.com/eiposgrados](https://instagram.com/eiposgrados)